IN THE SPECIFICATION:

Please amend the Specification as indicated below, where additions are indicated by underlining and deletions are indicated by strikethrough.

Please replace the paragraph beginning on page 1 at line 9 with the paragraph below.

This application claims the benefit of ~~priority to~~ United States Provisional Patent Application Serial No. 60/157,272 filed ~~01~~ on October, 1 1999 and entitled *Data Structures and Algorithms For Simulated Interaction With Graphical Objects*, which application is hereby incorporated by reference.

Please replace the paragraph beginning on page 2 at line 2 with the paragraph below.

Heretofore the ability to integrate real-time three-dimensional hand interaction into software applications has been ~~some-what~~ somewhat limited and has required a relatively high level of skill among practitioners. Complex three-dimensional worlds have not been easy to simulate and frequently such simulations ~~where~~ were performed in a rudimentary manner as the required results would not support the massive efforts needed to understand the simulated to environment, objects in that environment, or the input/output devices with interaction with the environment and objects would be made. ~~Nor~~ It also has ~~it~~ not been practical to ~~was time developing~~ develop complex scene graphs and mapping between the several scene graphs that might typically be present. System interaction and the visual and tactile/force feedback were also in need of improvement, particularly where it was desired that visual feedback cues be synchronized with tactile and force cues, especially for ~~had~~ hand grasping interactions of virtual objects.

Please replace the paragraph beginning on page 3 at line 16 with the paragraph below.

The invention provides a structure, method, computer program and computer program product for novel object simulation and interaction of and between computer generated or graphical objects in virtual space. These include novel Neutral Scene ~~Graphs~~ Graph data structures and procedures for using such graphs, Object-Manipulation Procedures, and impedance mode procedures and techniques for simulating physical interaction with computer generated graphical objects in virtual space.

Please replace the paragraph beginning on page 4 at line 17 with the paragraph below.

After these three aspects have been described in some detail, the implementation of one particular embodiment of the invention as computer software is described in considerable detail so that the interaction between the above described three aspects and other elements of a simulation and interaction system and method may more clearly be understood. This implementation is referred to as the Virtual hand Toolkit (VHT), which is a component of ~~the~~ a Virtual ~~hand~~ Hand Suite ~~2000~~.

Please delete the title appearing on page 5 at lines 3 and 4 as shown below.

~~**Description of Embodiments of Neutral Scene Graphs, Object Manipulation, and Impedance Mode Techniques**~~

Please replace the paragraph beginning on page 7 at line 8 with the paragraph below.

In one exemplary embodiment, illustrated in FIG. 1, one NSG synchronizes two scene graphs, scene graph A (SG-A) and scene graph B (SG-B), and each node in the NSG (nodes shown as square ~~box~~ boxes) contains two pointers, one to a node in graph A (nodes shown as round circles) and one to a node in graph B (nodes shown as triangles). NSG pointers are also illustrated as are scene graph ~~edged~~ edges using different ~~arrow-heads~~ arrowheads as indicated in the ~~figure~~ key shown in FIG. 1.

Please replace the paragraph beginning on page 8 at line 6 with the paragraph below.

The Neutral Scene Graph (NSG) described here is particularly useful since typically computer graphics associated with a 2D or 3D simulation only ~~needs~~ need to be updated from between about 10 times per second to about 60 times per second since these rates are typically sufficient to support visual observation. In contrast, a haptic (i.e., force feedback) display and the associated interactive mathematical model of the simulation typically requires (or at least greatly benefits from) much faster update, such as more than 100 times per second and typically from about 1000 times per second to 2000 times per second or more. Thus, the graphical update (e.g. 10-60 times/second update frequency) can advantageously be run or executed in a separate computer program software thread, and at a much different update rate, from the corresponding haptic or interaction thread (e.g. 1000-2000 times/second update frequency). ~~Those workers in the art will appreciate that these~~ These data update frequencies are provided for the purpose of illustration and ~~that~~ the invention itself is not limited to any particular update rate or frequency. The inventive NSG provides an efficient mechanism to ensure that the updated graphical data remains synchronized with the corresponding updated haptic or interaction data, even if and when they are updated at different times and intervals.

Please amend the title appearing on page 9 at line 1 as shown below (leaving the original underlining).

~~Embodiment of~~ Object-Manipulation Structure and Procedure

Please replace the paragraph beginning on page 9 at line 11 with the paragraph below.

Each virtual object to be grasped (or otherwise contacted) has an associated MFSM that in turn has an associated grasping virtual object "parent," referred to as the GParent. (For simplicity of explanation, a grasping type contact will be assumed for purposes of description here, but the invention is not limited only to grasping type contact.) A GParent is a virtual object that is capable of grasping another virtual object. A GParent may be any virtual object independent of a virtual object to be grasped, and each GParent virtual object may have any number of other virtual objects using it as a GParent. Collision information between a virtual object to be grasped and its GParent is used in the inventive algorithm, method, and computer program. The state of the manipulation finite-state machine (MFSM) determines how a virtual object behaves with respect to ~~it~~ its GParent.

Please replace the paragraph beginning on page 10 at line 12 with the paragraph below.

Transition of and between the states in the exemplary embodiment of the MFSM of FIG. 2 ~~are~~ is governed by various parameters including but not limited to the following parameters: minimum normal angle (MinNormalAngle), minimum drop angle (MinDropAngle), number of drop fingers (NumDropFingers), and grasp one cone slope (GraspOneConeSlope).

Please replace the paragraph beginning on page 12 at line 9 with the paragraph below.

With further reference to the state diagram of FIG. 2, the diagram illustrates possible state transitions between the states of the MFSM that can occur during the state recalculation. These transitions include a transition from NONE to ONE in response to a "new contact" condition, and a transition between ONE and NONE in response to a "0 contact" (zero contact) condition. New contact is the condition when a point on the GParent, which in the previous iteration was not in contact with the virtual object to be grasped, comes into contact with the virtual object. Zero Contact (or no contact) is the condition when in the previous algorithm iteration there was at least one point on the GParent in contact with the virtual object to be grasped, but there are no longer any such contact points. The manipulation state also transitions from RELEASE to NONE in response to the "0 contact" condition. The manipulation ~~Manipulation~~ state transitions from the ONE state to the G3 state when there are a good set of normals and three contacts. Normals are considered to be good when there is at least one pair of contract normals which have an angle between them exceeding the MinNormalAngle parameter. The manipulation state in the MSFM transitions from G3 to RELEASE when the number of "manipulators" which have an angle less than the minimum drop angle is not greater than or equal to NumDropFingers. An example value for NumDropFingers is 2. Once the state machine is in the ONE state, it will remain in that state when any new contact condition occurs, unless conditions are met to transition to the G3 state.

Please replace the paragraph beginning on page 14 at line 14 with the paragraph below.

For each collision that is detected, the collision information, including (1) the contact point between the manipulator (i.e., the object pointed to by GParent, which is sometimes referred to as the Virtual End Effector or VEE) and the virtual object, (2) the normal to the contact point, and the like, are sent to the MFSM corresponding to the object that is colliding with the manipulator (Step 108). Finally, the grasping state for all MFSMs are recalculated at the interaction update rate, which is typically as fast as the interaction software thread can run. Exemplary pseudo code for the grasping algorithm is set forth immediately below:

Please amend the title appearing on page 15 at line 16 as shown below (leaving the original underlining).

~~Embodiment of~~ Impedance Mode Structure and Procedure

Please replace the paragraph beginning on page 15 at line 17 with the paragraph below.

In yet another aspect, ~~the subject invention provides~~ a haptic-control technique for low-bandwidth updates of a force-feedback controller and ~~a force-feedback controller and~~ method of control incorporating or utilizing these low-bandwidth haptic-control updates. This ~~inventive~~ technique is particularly applicable to situations where a simulation is executing asynchronously (either on the same computer in a different thread or on a separate computer) from an associated force-control process.

Please replace the paragraph beginning on page 16 at line 15 with the paragraph below.

A ~~An~~ Local Surface Approximation (LSA) includes an approximation of at least one, and advantageously~~,~~ each, of the factors in a region near the VEE that are likely to influence the VEE. The LSA description may, for example, include information about the geometry, velocity, acceleration, compliance, texture, temperature, color, smell, taste, and the like, or any other physical, static, ~~dynamical~~ dynamic or any other property associated with one or more nearby virtual objects. A geometric approximation may, for example, include: (1) a single polygon (such as a virtual object tangent plane), (2) an approximating polyhedron, (3) a parametric surface (such as NURBS or subdivision surfaces), and the like.

Please replace the paragraph beginning on page 17 at line 9 with the paragraph below.

Each VEE in the simulation may have multiple possible ~~LSA's~~ LSAs associated with it due to the multiple virtual objects in a simulation. Two ~~exemplary~~ embodiments of procedures for associating LSAs with VEEs are now described. ~~In the first possible procedure,~~ First, the LSAs are sorted by contact state with assignment to a VEE having the LSA closest to it. ~~In a second possible procedure,~~ Second, a new composite LSA that encompasses all ~~LSA's~~ LSAs within a fixed distance of the VEE is constructed.

Please replace the paragraph beginning on page 17 at line 17 with the paragraph below.

One embodiment of the host-computer-side procedure and algorithm is now described. While the simulation is running or executing on a general purpose computer, collision detection checks are performed between the VEE and the virtual environment (Step 122). In one embodiment of the invention, collision detection may ~~utilized~~ utilize the techniques described in co-pending United States Utility Application Serial no. 09/432,362 filed 11/3/99 and entitled "System And Method For Constraining A Graphical Hand From Penetrating Simulated Graphical Objects," which is ~~hereby~~ incorporated by reference above.

Please replace the paragraph beginning on page 18 at line 1 with the paragraph below.

The paragraph above describes the collision-check step~~, now the~~. Subsequently, ~~remaining steps follow: Construct~~ a set of LSAs is constructed for each potential collision between the VEEs and all the virtual objects. Each LSA in the set corresponds to the object portion that is in contact or potential contact with the VEEs (See FIG. 3). The set of LSAs are stored in a memory buffer.

Please replace the paragraph beginning on page 18 at line 8 with the paragraph below.

For each VEE, ~~find~~ the subset of LSAs in the memory buffer that correspond to that particular VEE is found and ~~. Perform~~ one of the following two steps is performed: (i) combine the subset of ~~LSA~~ LSAs into a single set, and ~~)ii)~~ (ii) select the most significant LSA out of the subset. In either case, one LSA is constructed ~~of~~ <u>or</u> selected for each VEE. The final step is to send this LSA to the PEE controller.

Please replace the paragraph beginning on page 19 at line 14 with the paragraph below.

The fast-thread may employ a variety of control ~~algorithm~~ <u>algorithms</u> including control algorithms known in the art and proprietary control algorithms. Such control algorithms, may for example include position control and force control <u>algorithms</u>. For example, for position control, the force applied by the PEE is determined based on the discrepancy between desired position and/or velocity of the VEE with respect to the haptic surface texture. For force control, the force applied by the PEE is determined based on the discrepancy between applied and computed forces.

Please replace the paragraph beginning on page 20 at line 14 with the paragraph below.

An exemplary embodiment of the Physical End Effector (PEE) side slow-thread and fast-thread loop control procedures ~~are~~ <u>is</u> summarized in pseudo-code immediately below.

Please amend the title appearing on page 22 at line 1 as shown below (leaving the original underlining).

~~Embodiment of~~ Virtual Hand Toolkit (VHT) and VirtualHand Suite

Please replace the paragraph beginning on page 22 at line 2 with the paragraph below.

The Virtual Hand Toolkit (VHT) is the application development component of the VirtualHand Suite ~~2000~~ <u>(VHS)</u>, which also includes ~~the~~ <u>a</u> Device Configuration Utility and the Device Manager. The later two components are described in detail in the VirtualHand Suite User's Guide. ~~These technology is developed by Virtual Technology, Inc. of Palo Alto, California.~~

Please replace the paragraph beginning on page 22 at line 7 with the paragraph below.

~~Virtual Technology Inc (Vti) hardware will typically ship with basic software~~ Hardware can be used to help users access devices~~.~~; however, ~~purchasing~~ the VirtualHand Suite offers significant additional functionality, including the complete set of libraries included in the VHT. The goal of VHT is to help developers easily integrate real-time 3D hand interaction into their software applications, ~~thus~~ minimizing development efforts. Complex 3D simulated worlds are not easy to implement, and programmers ~~can't afford to~~ sometimes cannot spend the time required to understand all the inner workings of advanced input/output devices such as ~~the~~ a CyberGlove, CyberTouch, or ~~and~~ CyberGrasp device. ~~Nor can they~~ Programmers also sometimes cannot waste time on other complex issues such as collision-detection, haptic scene graphs, event models and global system response.

Please replace the paragraph beginning on page 23 at line 3 with the paragraph below.

The VHT is transparent to the programmer's work, as its core is based on a multi-threaded architecture ~~backed~~ with event-based synchronization. This is similar to GUI-based applications where the programmer need not manage the mouse or service graphical user interface (GUI) events. The application development process should focus on application-specific functionality, not low-level details. To develop an application that makes use of ~~Virtual Technologies'~~ whole-hand input devices, according to an embodiment of the invention, the ~~softwar~~ software designer is provided with an application model that consists of three major components:

- A ~~Virtual~~ human hand;
- An ~~Object~~ manipulation and interaction; and
- A ~~Rendering~~ capability.

Please replace the paragraph beginning on page 23 at line 14 with the paragraph below.

The Virtual Hand Toolkit (VHT) is divided into a number of functional components~~. This fact is reflected in the~~ corresponding to a division of ~~the~~ libraries. According to one or more embodiments, the ~~The~~ toolkit is ~~now~~ implemented by using two libraries and a set of support libraries.

Please replace the paragraph beginning on page 23 at line 17 with the paragraph below.

Purchased ~~All purchased VTi~~ hardware can come with ~~comes~~ basic software that includes ~~the~~ Device Configuration Utility, Device Manager and the Device layer ~~part~~ components of the Virtual Hand Toolkit. This can include ~~includes VTi~~ hardware support, for example, in the form of device proxy classes as well as a set of classes for doing 3D math ~~and finally~~ as well as a set of exception classes. The basic device layer is described in detail ~~in Chapter 4~~ below.

Please replace the paragraph beginning on page 24 at line 1 with the paragraph below.

The main functionality of the ~~VHTis~~ VHT is contained in the Core layer and includes both the haptic and neutral scene graphs, simulation support, collision interface, model import, human hand support (including grasping and ghosting).

Please replace the paragraph beginning on page 24 at line 4 with the paragraph below.

Third-party support packages and interfaces ~~are~~ can be located, in separate specific libraries. For example, support for ~~the~~ a Cosmo/Optimizer display engine ~~and import~~ can be imported into the VHT ~~is located~~ and stored in a separate library. Also, local geometry level collision detection is externalized and the VHS includes two modules that may be used. The relationships between these levels can be seen in FIG. 4.

Please replace the paragraph beginning on page 24 at line 11 with the paragraph below.

A complete working example is also presented ~~throughout the chapters, for a hands- on experience of application development~~ in detail below.

Please replace the paragraph beginning on page 24 at line 13 with the paragraph below.

~~Those workers having ordinary skill skilled in the art are assumed to have a~~ A working understanding of C++ programming language and ~~are proficient~~ proficiency with a development environment on their platform (such as Windows NT or SGI IRIX) is assumed. Some basic knowledge of 3D graphics is also assumed. Additional background information to provide or supplement this knowledge is provided in standard textbooks such as: ***Computer Graphics, Principles and Practice***, J. Foley, A. van Dam, S. Feiner, J. Hughes. 2nd Edition, Addison-Wesley, 1996; and ***The C++ Programming Language***, Bjame Stroustrup, 3rd Edition, Addison-Wesley, 1997; each of which is hereby incorporated by reference. ~~In this section, an overview of the Virtual Hand Toolkit (VHT) is presented. Simple examples are used to illustrate the usage of the main classes in the VHT. Furthermore, the proper development environment settings (for NT/2000 and IRIX) are explained. For a fuller understanding and discussion of the classes, please refer to the subsequent sections (chapters).~~

Please add the following paragraph on page 25 at line 4 as shown below.

According to an embodiment of the invention, an overview of the Virtual Hand Toolkit (VHT) is presented using simple examples to illustrate the usage of the main classes in the VHT. Furthermore, the development environment settings (for NT/2000 and IRIX) are explained. Additional detail regarding the classes is provided below.

Please replace the paragraph beginning on page 25 at line 6 with the paragraph below.

In an application, the Virtual Human Hand class (vhtHumanHand) is the high-level front-end for ~~Virtual Technologies'~~ whole-hand input devices. It lets the developer easily specify what kind of devices are to be operated, and where they are serviced. Then it takes care of all further operations automatically. The aim is to make these advanced I/O devices as transparent as possible, much like the computer mouse in traditional applications.

Please replace the paragraph beginning on page 25 at line 14 with the paragraph below.

Developing an interactive 3D application is a complex process because of the necessity of defining simulated objects and controlling their state at run-time. The VHT ~~TheVHT~~ has a significant section dedicated to these particular tasks. The Haptic Hierarchy and associated import filters provide a simple way to specify the geometrical and physical details of digital objects. The Collision Detection Engine keeps track of any contact between objects as they move or change shape and provides the simulation environment with this information in a multi-threaded fashion.

Please replace the paragraph beginning on page 26 at line 2 with the paragraph below.

Some ~~Most~~ applications need to present the mathematical and geometrical information discussed ~~in the previous sections~~ above in a more visual form, namely as 3D digital objects. Although that particular task does not involve the VHT, close cooperation is required to achieve realistic rendering. For this the VHT provides a Data-Neutral Scene Graph that binds user-specific data with haptic objects for synchronization, or notification of haptic events. It also offers the functionality to render digital hands that closely mimic the behaviour of a human hand being measured by ~~a VTi~~ an instrumented glove such as the CyberGlove.

Please replace the paragraph beginning on page 26 at line 11 with the paragraph below.

There are three simple steps to follow when using the VHT as the main haptic simulation loop of your application. ~~In this guide, we present the most common~~ One approach is: connecting to a CyberGlove and a six degree-of-freedom (6-DOF) position tracker (Polhemus or Ascension), associating the devices to the application's virtual hand (vhtHumanHand instance), and creating the support environment with the vhtEngine and vhtSimulation classes. ~~If you have not already done so, you should refer to the VirtualHand Suite User's Guide as well as the hardware manuals before proceeding further.~~

Please replace the paragraph beginning on page 26 at line 20 with the paragraph below.

The Virtual Hand Toolkit uses a client/server architecture in which the user's application acts as the client. The physical devices reside on the server side, also known as the Device Manager. The Device Manager can be running on the same machine as ~~you're~~ a user's application, or on any other machine on ~~your~~ a network (or even the Internet~~, although performance may suffer~~). In either case, it is necessary to specify which of the devices are going to be used by the application.

Please replace the paragraph beginning on page 27 at line 5 with the paragraph below.

The address of a CyberGlove is given through a helper class named vhtIOConn. The straightforward use of vhtIOConn is through ~~VTi's~~ a resource registry, which is specified in an external file. If the ~~VTI_REGISTRY_FILE~~ REGISTRY_FILE environment variable specifies a registry file, then to attach to the default glove you only have to ~~do~~ set the variable as shown below:

Please replace the paragraph beginning on page 27 at line 11 with the paragraph below.

One can also provide all the details in ~~your~~ software code. ~~We will use as an~~ For example, a setup in which ~~the~~ a Device Manager is running on the same machine as the application (*localhost*), the device being used is a CyberGlove (*cyberglove*) and it is connected on the first serial port (*COM1*) and running at maximum speed (*115200 baud*).

Please replace the paragraph beginning on page 27 at line 18 with the paragraph below.

The third parameter in the gloveAddress specification is the port number of the Device Manager which by default is 12345. ~~Should you encounter~~ If problems connecting are encountered, ~~you should contact~~ the person who installed the Device Manager should be notified ~~to know if it is expecting connections on a different~~ to determine the proper port number.

Please replace the paragraph beginning on page 28 at line 10 with the paragraph below.

~~We can also specify which~~ Which of the tracker's position receivers to use (some have more than one) can also be specified. An individual tracker receiver is supplied by the vhtTracker::getLogicalDevice method, which returns a vht6DofDevice instance. Thus, the following code segment gets the first receiver on the tracker, and associates it with the glove defined previously through the helper class vhtHandMaster:

Please replace the paragraph beginning on page 28 at line 20 with the paragraph below.

At this point, the vhtHumanHand object is ready to be used as a data-acquisition device. ~~We are interested in using it~~ This object can be used at a higher-level~~, so~~ by using the next step ~~is~~ to set up a more elaborate environment.

Please replace the paragraph beginning on page 29 at line 2 with the paragraph below.

The vhtEngine is a central container for the VHT runtime context. For normal operations, ~~you will create~~ a single instance of the vhtEngine class can be created in a user's ~~your~~ application, ~~like this~~ using the following line of code:

Please replace the paragraph beginning on page 29 at line 15 with the paragraph below.

The two previous steps created a transparent infrastructure for an application that will perform hand-based manipulation. The next step is to supply the extension path through which the application will interact with the VHT and provide it with the simulation logic. The VHT requires that some kind of simulation logic be registered with the vhtEngine. For the most basic cases, ~~you can use~~ the vhtSimulation class can be used by registering~~. You register~~ a simulation object in the vhtEngine instance as follows:

Please replace the paragraph beginning on page 30 at line 10 with the paragraph below.

The haptic scene graph can be constructed by specifying the geometry of simple objects from within the application. Given the complexity of the average digital object, applications ~~will most likely~~ can import object descriptions from data files generated by third-party 3D modellers.

Please replace the paragraph beginning on page 30 at line 10 with the paragraph below.

The haptic scene graph can be constructed by specifying the geometry of simple objects from within the application. Given the complexity of the average digital object, applications ~~will most likely~~ can import object descriptions from data files generated by third-party 3D modellers.

Please replace the paragraph beginning on page 31 at line 1 with the paragraph below.

When analytical surfaces are required, one can use the elementary common shapes by instantiating objects using the vhtVertexBox, vhtVertexSphere and other basic shape classes. While these geometries are not as common in real-world simulations, they offer the advantage of being well-defined. Although, ~~this current version of~~ according to an embodiment, the VHT does not perform any optimization on these surfaces, ~~future versions will~~ other embodiments take advantage of their known mathematical properties to process operations faster than with general polyhedra.

Please replace the paragraph beginning on page 31 at line 10 with the paragraph below.

One ~~maycreate~~ can create a polyhedron object by defining a set of vertices and assigning them, with the vertex count, to a vhtVertexGeometry. The resulting polyhedron is then assigned to a vhtShape3D instance. For a simple unit cube, this is done in the following fashion:

Please replace the paragraph beginning on page 32 at line 11 with the paragraph below.

Note that this procedure actually creates a geometry template for the collision engine your application will use. This will be discussed further ~~in the section on collisions~~ below.

Please replace the paragraph beginning on page 33 at line 17 with the paragraph below.

~~The scope of the VHT does not cover the actual simulation of digital objects. The~~ A developer ~~is responsible for creating~~ can use his or her own digital environment. The role of the VHT is to make it easy to "hand enable" such simulations. ~~However, the VHT does provide an expansion path which aims to aid the user with the implementation of simulation logic.~~

Please replace the paragraph beginning on page 34 at line 11 with the paragraph below.

As an example, ~~we will create the~~ simulation logic for a cube that spins on itself is ~~. It will be~~ provided by the class UserSimulation, which has the following declaration:

Please replace the paragraph beginning on page 36 at line 4 with the paragraph below.

A substantial requirement of any interactive simulation is to recreate physical presence and to give virtual objects the ability to be more than mere graphical shapes. ~~The goal is to have them~~ The virtual objects move in digital space and react to the collisions that may occur. The collision detection engine is normally used by a to subclass of vhtSimulation to detect these situations and then act upon them (during the handleConstraints phase). The following is only a brief overview of the collision detection capabilities of the VHT~~, and you should refer to the collision detection example in Chapter 8 to learn how to use it effectively~~.

Please replace the paragraph beginning on page 39 at line 2 with the paragraph below.

The previous sections have dealt with the more invisible parts of an application. ~~We have now reached the point where it's time to actually start seeing~~ According to one or more embodiments of the invention, it is possible to see the result of all those computations ~~that take effect~~ in the application. ~~As for the simulation logic, the scope of the VHT doesn't cover 3D rendering. However, since it is such an important operation,~~ According to one or more embodiments of the invention, the VHT ~~does~~ can provide an expansion path ~~for making it easy to draw~~ to facilitate drawing or creating virtual objects.

Please replace the paragraph beginning on page 39 at line 15 with the paragraph below.

Yet another graph, the *data neutral* graph (*e.g.*, the Neutral Scene Graph or NSG), will provide the glue between the two scene graphs. Typically, the data neutral graph is created as objects are imported and transferred into the haptic scene graph, as mentioned ~~in the section~~ under the heading *Using a NodeParser* below. The data neutral scene graph mechanism is explained in ~~Chapter 5~~ greater detail below.

Please replace the paragraph beginning on page 39 at line 15 with the paragraph below.

Yet another graph, the *data neutral* graph (*e.g.*, the Neutral Scene Graph or NSG), will provide the glue between the two scene graphs. Typically, the data neutral graph is created as objects are imported and transferred into the haptic scene graph, as mentioned ~~in the section~~

under the heading *Using a NodeParser*. The data neutral scene graph mechanism is explained in ~~Chapter 5~~ greater detail below.

Please replace the paragraph beginning on page 40 at line 10 with the paragraph below.

In the following code segment, the first element of the list of data neutral nodes associated with vhtComponents is queried. Then, a loop iterates through the elements of the list. At each iteration, it extracts the CosmoCode node and its vhtComponent equivalent (if any), and copies the transformation of the vhtComponent into the CosmoCode node's matrix; using the correct ordering of matrix elements. Note that this example is based on a utilization of the vhtCosmoParser class, which ~~always~~ can create a map between vhtComponent and csTransform instances. For this reason, no class checking is implemented in the loop, but in general such a blind operation would be error-prone.

Please replace the paragraph beginning on page 41 at line 23 with the paragraph below.

The VHT device layer is included with ~~all VTi~~ some hardware products, according to one or more embodiments of the invention. It contains a set of classes to facilitate access to input devices. These devices include CyberGlove, CyberTouch and CyberGrasp, as well as Polhemus and Ascension 6-DOF trackers, for example. Since, according to an embodiment of the invention, ~~all~~ direct hardware interaction ~~is~~ can be done by the Device Manager, instances of these classes provide a proxy representation of hardware devices. The proxy mode is distributed, so hardware may even be at remote locations and accessed, for example, via a TCP / IP network.

Please replace the paragraph beginning on page 42 at line 9 with the paragraph below.

Some ~~Most~~ applications will build vhtIOConn objects by referring to predefined entries in the device registry, which is maintained by the DCU. ~~(see the VHS User's Guide for more details about the DCU).~~ To access a default device defined in the registry, the application code will use the vhtl0Conn::getDefault method. For example, the statement to get the glove proxy address using the registry defaults is:

Please replace the paragraph beginning on page 43 at line 9 with the paragraph below.

According to one of more embodiments of the invention, ~~Note that~~ the VHT scene graphs are primarily oriented toward haptic and dynamic operations, rather than graphic rendering.

Please replace the paragraph beginning on page 44 at line 12 with the paragraph below.

According to one or more embodiments of the invention, ~~Finally, it should be noted that from a theoretical standpoint;~~ the VHT haptic scene graph is a directed tree without cycles, rather than a general graph. According other embodiments of the invention, however, ~~This~~ this limitation might be removed ~~in the future~~.

Please replace the paragraph beginning on page 44 at line 15 with the paragraph below.

~~From the above discussion, you can see that any useful~~ As discussed above, a scene graph ~~will generally~~ can contain at least two types of nodes, namely transformation grouping nodes and geometric nodes. By organizing these two types of nodes into a tree-like data structure, we obtain a hierarchy of geometric transformations that can be applied ~~on atomic geometrical~~ to geometric shapes. In the VHT, the vhtTransformGroup and vhtShape3D classes provide the basic set of scene graph nodes. The haptic scene graph can be constructed by inserting nodes one by one from method calls, or by using a model parser like the vhtCosmoParser.

Please replace the paragraph beginning on page 45 at line 19 with the paragraph below.

To create hierarchies of nodes, ~~we need to be able to define~~ the relationships between the nodes can be defined. This is accomplished with the help of the addChild(vhtNode *nPtr) method of the vhtTransformGroup class. To build a simple hierarchy with two levels, one of which is translated 10 units along the ~~x axis~~ x-axis, ~~we could write~~ the following can be used:

Please replace the paragraph beginning on page 47 at line 10 with the paragraph below.

The VHT ~~currently includes~~ can include pre-defined classes ~~only~~ for convex vertex-based geometries, according to one or more embodiments of the invention; however,~~. However,~~ users ~~are free to~~ can also extend the geometry classes to suit their need. Once a vhtShape3D object has been created, it can be added as a child to a vhtTransformGroup node (a vhtShape3D can be the child of only one grouping node a the time). Thus, to translate ~~our~~ a unit cube 10 units in the x direction, we could reuse the vhtTransformGroup variable introduced in the previous code example as ~~such~~ shown below:

Please replace the paragraph beginning on page 48 at line 3 with the paragraph below.

~~As a closing note to this section, it should be noted that every~~ According to one or more embodiments of the invention, each type of node class of the VHT haptic scene graph ~~is~~ can be equipped with a render method. By invoking this method, ~~you get~~ a visual representation of the node sent into the current OpenGL context can be obtained. For vhtShape3D nodes, the render method simply calls the render method of the first vhtGeometry that has been set. Because ~~Since~~ the VHT, according to one or more embodiments of the invention, ~~by default only~~ primarily includes support for ~~vertex based~~ vertex-based geometries, this method will render a point cloud for each vhtShape3D. The geometry rendered can be manipulated by defining an appropriate vhtCollisionFactory framework. ~~Note that if~~ If a user application uses the VClip interface, however, the rendered geometry ~~will~~ can contain face information (i.e. solid shaded geometries), according to one or more embodiments..

Please replace the paragraph beginning on page 48 at line 21 with the paragraph below.

~~In the previous section, the~~ The scene graph, although not described above as ~~was not~~ being modified after it has been constructed, can be so modified. ~~But most~~ Indeed, some user applications, according to one or more embodiments of the invention, ~~will~~ require that objects in the virtual environment move in some manner. Using ~~Since we have~~ access to the transform of a vhtTransform3D node, ~~it is fairly clear that we can just modify~~ the coordinate frame of each geometry can be modified directly. Using the above code samples, ~~we could add~~ the following line of code could be used to create motion:

Please replace the paragraph beginning on page 49 at line 6 with the paragraph below.

That statement displaces the cube from its original position (10,0,0) to the point (11,0,0). If ~~we do~~ this statement is executed in a loop that draws the haptic scene graph, once per frame, the cube will seem to move along the x-axis at a fairly high rate of speed. ~~However, there is one detail that prevents this from working properly.~~ The render method of all haptic scene graph nodes, however, uses ~~the~~ each node's LM matrix for OpenGL, which ~~have to~~ can be synchronized with the changes caused to a node. ~~In order to~~ To keep the transform and the LM in synch, ~~it is necessary to call~~ the refresh method can be called, for example. Refresh is a recursive method that works on all nodes in a sub graph ~~so it needs~~ and only needs to be called on the root node, according to an embodiment, and can, for example, ~~. Primarily, refresh~~ will ensure that all LM's and transform matrices agree with each other. Thus, ~~we need to add one more~~ the following statement can be added to the previous one: